REMARKS

Applicant respectfully requests reconsideration and allowance of the subject application. Claims 13-24 are pending, of which claims 13-23 have been amended.

35 U.S.C. §101 Claim Rejections

Claims 13-24 are rejected under 35 U.S.C. §101 as being directed to non-statutory subject matter (*Office Action* p.3). Appropriate amendments to claims 13-23 have been provided herein and Applicant respectfully requests that the §101 rejection be withdrawn.

35 U.S.C. §102 Claim Rejections

Claims 13-24 are rejected under 35 U.S.C. §102(b) as being anticipated by Hendren et al., "Supporting Array Dependence Testing for an Optimizing/ Parallelizing *C* Compiler" (1993) (hereinafter, "Hendren") (*Office Action* p.5). Applicant respectfully traverses the rejection.

As the title indicates, Hendren describes features for a *C* compiler, such as support analyses in the context of *C* parallelizing compilers to handle loops and aliasing due to pointers (*Hendren* p.2, ¶1). Specifically, Hendren describes a framework designed to handle the complexities of the *C* language that affect array dependence analysis (*Hendren* p.3, ¶1.2). Hendren describes, and illustrates in Fig. 2, that *C* program files are input to a compiler which produces a simplified

and compositional structured representation called "SIMPLE" to handle various complications in the *C* program files (*Hendren* p.4, ¶2; Fig. 2).

In the Background section of the present application, Applicant describes compiling a high-level programming language into machine instructions where the compiling includes lexical, syntax, and semantic analysis, as well as generating a syntax tree during compiling (*Specification* p.1, line 19 to p.2, line 18). In contrast to a compiler that translates a program written in a *specific* programming language (e.g., the *C* programming language) into another programming language or machine code, as described in Hendren, Applicant describes that a program can be developed and represented by a "high-level program tree that is a *syntax-independent* representation" of a programmer's intent (*Specification* p.4, lines 33-37). A programmer can directly manipulate the syntax-independent program tree, "which is in contrast to conventional programming systems in which a programmer manipulates a textual representation of the program that is later converted into a syntax tree during compilation" – as described in Hendren. (*Specification* p.5, lines 1-5).

Claim 13 recites a method comprising "receiving multiple nodes of a program tree as user-selected node inputs, each of the nodes specifying a declaration pointer and an operand pointer to another node", and "storing the multiple nodes of the program tree that is a syntax-independent representation of a computer program such that the declaration pointers and the operand pointers interconnect the nodes to form the program tree".

Hendren does not show or disclose receiving multiple nodes of a program tree as user-selected node inputs, or that a program tree is a syntax-independent representation of a computer program, as recited in claim 13.

As described above, Hendren is syntax-dependent and describes an intermediate structure to compile *C* program files (*Hendren* p.4, ¶2; Fig. 2). The Office cites Hendren for teaching a syntax-independent programming intent represented as a first node of a data structure (*Office Action* p.4). Applicant disagrees because what Hendren illustrates in Fig. 7(b) is a general expression tree that represents an index in an example simplified program illustrated in Fig. 7(a) (*Hendren* pp.9-10, ¶5.1; Fig. 7).

Hendren describes that a general expression tree is built during the structured "SIMPLE" analysis when compiling the *C* program files (*Hendren* p.4, ¶2; Fig. 2; p.9, ¶5.1). Nodes of the general expression tree in Hendren are not received as user-selected node inputs of a program tree, as recited in claim 13. As described in Hendren, the general expression tree is generated from a source program during compiling. Contrary to Hendren, Applicant claims a program tree that is a syntax-independent representation of a computer program.

Accordingly, claim 13 along with dependent claims 14-16 and 21-24 are allowable over Hendren and Applicant respectfully requests that the §102 rejection be withdrawn.

Claim 17 recites a method of handling data comprising "receiving multiple nodes of a hierarchical tree that is a syntax-independent representation of a computer program, each of the nodes received as user-selected node inputs that specify a declaration pointer and an operand pointer to another node".

Hendren does not show or disclose receiving multiple nodes of a hierarchical tree as user-selected node inputs, or a hierarchical tree that is a syntax-independent representation of a computer program, as recited in claim 17.

As described above in the response to the rejection of claim 13, Hendren is syntax-dependent and describes an intermediate structure to compile *C* program files (*Hendren* p.4, ¶2; Fig. 2). Further, Hendren describes that a compiler generates a general expression tree from a source program during compiling (*Hendren* p.4, ¶2; Fig. 2; pp.9-10, ¶5.1). There is no indication in Hendren that nodes of the general expression tree are received as user-selected node inputs, as recited in claim 17.

Accordingly, claim 17 along with dependent claims 18-20 are allowable over Hendren and Applicant respectfully requests that the §102 rejection be withdrawn.

Conclusion

Pending claims 13-24 are in condition for allowance. Applicant respectfully requests reconsideration and issuance of the subject application. If any issues remain that preclude issuance of this application, the Examiner is urged to contact the undersigned attorney before issuing a subsequent Action.

Respectfully Submitted,

Dated: Jan 4, 2005 By: _____

David A. Morasch
Reg. No. 42,905
(509) 324-9256 x 210